INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team proval

Proofs of programs

Saclay - Île-de-France

Theme : Programs, Verification and Proofs

Table of contents

1. Team **1**
2. Overall Objectives **1**
2.1. Introduction 1
2.2. Highlights 2
3. Scientific Foundations **3**
3.1. Higher-Order Functional Languages 3
3.1.1. Randomized algorithms 3
3.1.2. Floating-point programs 3
3.1.3. Certification of tools 3
3.2. Proof of Imperative and Object-Oriented programs 4
3.2.1. The Why platform 5
3.2.2. Applications and case studies 5
3.3. Automated deduction 6
3.3.1. Termination 6
3.3.2. Decision Procedures 6
3.3.2.1. Combination 6
3.3.2.2. Polymorphic Logics 6
3.3.2.3. The Alt-Ergo theorem prover 6
3.3.3. Automated proofs and certificates 7
3.4. Synchronous Programming 7
3.4.1. Extended synchronous models 8
3.4.2. The semantics of hybrid system modelers 8
4. Application Domains **8**
5. Software **9**
5.1. The CiME rewrite toolbox 9
5.2. The Why platform 9
5.3. The Alt-Ergo theorem prover 9
5.4. Lucid Synchrone 10
5.5. Reactive ML 10
5.6. Bibtex2html 10
5.7. Ocamlgraph 11
5.8. Mlpost 11
5.9. The Flocq library 11
5.10. The Gappa tool 11
5.11. The Interval package for Coq 11
5.12. The Alea library for randomized algorithms 12
5.13. The Coccinelle library for term rewriting 12
6. New Results **12**
6.1. Floating-Point Programs 12
6.2. Models and Proofs of Imperative Programs 13
6.3. Automatic Generation of Specifications 14
6.4. Certification 14
6.5. Contributions to functional programming environments 14
6.6. Synchronous Programming 14
6.6.1. The n-synchronous model 15
6.6.2. The semantics of hybrid system modelers 15
6.6.3. Dynamic aspects in synchronous languages 15
7. Contracts and Grants with Industry **16**
7.1. System@tic: Hi-Lite 16

7.2. CEA-Airbus contract 16
7.3. Airbus contract 16
8. Other Grants and Activities **16**
8.1. Regional Initiatives 16
8.1.1. Hisseo 16
8.1.2. Pactole 16
8.2. National initiatives 17
8.2.1. U3CAT 17
8.2.2. INRIA ADT Alt-Ergo 17
8.2.3. FOST 17
8.2.4. SIESTA 18
8.2.5. GENCOD 18
8.2.6. INRIA Action d'Envergure Synchronics 18
8.2.7. SCALP 18
8.2.8. PARTOUT 18
8.2.9. DECERT 19
8.3. European initiatives 19
8.4. Exterior research visitors 19
8.4.1. Visits 19
8.4.2. Invitations 19
9. Dissemination **20**
9.1. Interaction with the scientific community 20
9.1.1. Prices and distinctions 20
9.1.2. Collective responsibilities within INRIA 20
9.1.3. Collective responsibilities outside INRIA 20
9.1.4. Event organization 21
9.1.5. Editorial boards 21
9.1.6. Program committees 21
9.1.7. Invited Presentations 21
9.1.8. Theses defended 22
9.1.9. Participation to thesis juries 22
9.2. Teaching 22
9.2.1. Supervision of PhDs 22
9.2.2. Supervision of Post-docs and internships 23
9.2.3. Graduate courses 23
9.2.4. Other Courses 23
9.3. Industrial Dissemination 23
9.4. Popularization 24
10. Bibliography **24**

The Proval project-team is a research team common to INRIA - Saclay Île-de-France, CNRS and Université Paris-Sud 11. Researchers are also members of the LRI (Laboratoire de Recherche en Informatique, UMR 8623).

1. Team

Research Scientists

Claude Marché [Team Vice-Leader, Senior Researcher, HdR]
Sylvie Boldo [Junior Researcher]
Évelyne Contejean [Junior Researcher CNRS]
Jean-Christophe Filliâtre [Junior Researcher CNRS]
Guillaume Melquiond [Junior Researcher]

Faculty Members

Christine Paulin-Mohring [Team Leader, Professor University Paris-Sud 11, HdR]
Marc Pouzet [Professor University Paris-Sud 11 until March then ENS, member of IUF, HdR]
Sylvain Conchon [Associate Professor Université Paris-Sud 11, delegation INRIA]
Louis Mandel [Associate Professor Université Paris-Sud 11]
Andrei Paskevich [Associate Professor Université Paris-Sud 11]
Xavier Urbain [Associate Professor ENSIIE, Deleg. CNRS until Aug., Deleg. INRIA since Sep., HdR]

Technical Staff

Alain Mebsout [Junior Engineer]
Cécile Stentzel [Engineer]

PhD Students

Cédric Auger [University Paris-Sud 11]
Romain Bardou [University Paris-Sud 11]
François Bobot [University Paris-Sud 11]
Léonard Gérard [University Paris-Sud 11]
Paolo Herms [CEA grant]
Mohamed Iguernelala [University Paris-Sud 11]
Johannes Kanig [CORDI INRIA]
Stéphane Lescuyer [INRIA, on leave from X-Mines]
Thi-Minh-Tuyen Nguyen [INRIA Digiteo grant]
Asma Tafat Bouzid [University Paris-Sud 11]
Wendi Urribarrí [France-Venezuela scholarship, ATER University Paris-Sud 11 since Oct]

Post-Doctoral Fellows

David Baelde [Post-doc CNRS, since Sep]
Kalyan Krishnamani [Post-doc ANR grant]
Florence Plateau [ATER University Paris-Sud 11, until Nov]

Administrative Assistant

Régine Bricquet [TR]

Others

Cédric Pasteur [École Polytechnique, Master intern, March to August]
Simon Cruanes [École Polytechnique, undergraduate intern, May to July]

2. Overall Objectives

2.1. Introduction

Critical software applications in the domain of transportation, telecommunication or electronic transactions are put on the market within very short delays. In order to guarantee a dependable behavior, it is mandatory for a large part of the validation of the system to be done in a mechanical way.

The ProVal team addresses this question and consequently participates to the INRIA major scientific priorities:"Programming: Security and Reliability of Computing Systems".

Our approach uses *Type Theory* as a theoretical basis, a formalism which gives a clear semantics for representing, on a computer, both computation and deduction.

Type theory is a natural formalism for the specification and proof of *higher-order functional programs*, but we also use it as the kernel for *deductive verification of imperative programs*. It serves as a support for modeling activities (e.g. pointer programs, random computations, floating-point arithmetic, semantics).

Verification conditions (VCs) generated from programs annotated with specifications can often be expressed in simple formalisms (fragments of first-order logic) and consequently be solved using *automated deduction*. Building specialized tools for solving VCs, integrating different proof technologies, in particular interactive and automated ones, are important activities in our group.

When sophisticated tools are used for analyzing safety-critical code, their reliability is an important question: in an industrial setting, there is often a certification process. This certification is based on an informal satisfaction of development rules. We believe that decision procedures, compilers or verification condition generators (VCGs) should not act as black boxes but should be themselves specified and proved, or should produce evidence of the correctness of their output. This choice is influential in the design of our tools and is also a good challenge for them.

The project develops a generic environment (*Why*) for proving programs. *Why* generates sufficient conditions for a program to meet its expected behavior, that can be solved using interactive or automatic provers. On top of this tool, we have built dedicated environments for proving C (*Caduceus*) or Java (*Krakatoa*) programs.

With the arrival of Sylvie Boldo in 2005 and Guillaume Melquiond in 2008 as junior researchers, the team is developing a strong expertise in the area of formal verification of floating-point arithmetic.

Marc Pouzet joined the team as a full professor in September 2005, opening a research activity on synchronous systems. The goal is to propose high-level languages for the development of critical embedded systems with high temporal constraints. He obtained in March a new position at ENS. Members of ProVal working in this area consequentely moved their activity at this new location. They are currenly in the process of creating a new project-team.

Our research activities are detailed further, following the four themes:

- Higher-order functional languages,
- Proof of imperative and object-oriented programs,
- Automated deduction for program proof,
- Synchronous Programming.

Development of tools and applications is an important transversal activity for these four themes.

2.2. Highlights

As part of the FOST project, S. Boldo, F. Clément, J.-C. Filliâtre, M. Mayero, G. Melquiond and P. Weis studied a real-life program computing the discretization of the spread of acoustic waves on a rope. They developed a full formal proof of the method error of the simple three-point finite difference scheme for solving the 1D acoustic wave equation. An article describing this proof and the design choices for the operators (in particular the big O) has been published in the selective conference ITP (Interactive Theorem Proving 2010) [22]. This is to be joined with the corresponding rounding error that was also formally proved [1].

Our SMT theorem prover Alt-Ergo received a growing interest from critical software industry. Airbus France expressed in 2009 the wish to integrate Alt-Ergo in its process of certification of the critical softwares in their next generation planes. We thus started the procedure of *qualifying* Alt-Ergo in the sense of the DO-178B norm, which fixes the contraints on software development to achieve certification of an avionics software. This is done as part of the *Action de Developpement Technologique Alt-Ergo*. Alt-Ergo is also distributed by Altran/Praxis as part the last 2010 release of the SPARK/ADA verifier.

3. Scientific Foundations

3.1. Higher-Order Functional Languages

Participants: Sylvie Boldo, Évelyne Contejean, Jean-Christophe Filliâtre, Guillaume Melquiond, Christine Paulin-Mohring.

Higher-order strongly typed programming languages such as Objective Caml help improving the quality of software development. Static typing automatically detects possible execution errors. Higher-order functions, polymorphism, modules and functors are powerful tools for the development of generic reusable libraries. Our general goal is to enrich such a software environment with a language of annotations as well as libraries for datatypes, abstract notions and associated theorems which can express logical properties of programs and ease the possibility to automatically and interactively develop proofs of correctness of the programs.

In the past, we made contributions to the *Coq* proof assistant by adding functionalities for improving the development of formally proved functional programs. A first contribution is a new method to extract Ocaml modular code from *Coq* proofs (P. Letouzey PhD thesis [84], [85]). This extraction mechanism is an original feature for the *Coq* system, and has been used by several teams around the world in order to get efficient certified code [83]. Another contribution (M. Sozeau PhD thesis [94], [95]) is an extension of the *Coq* input language for building programs with strong specifications by writing only the computational part and generating separetely proof obligations (which are usually solved by tactics) and also a mechanism generalizing Type Classes à la Haskell which gives overloading in programs and proofs and facilitates the development of generic tactics..

We are using the capability of the *Coq* system to model both computation and deduction in order to explore different classes of applications. These examples involve the development of large reusable *Coq* libraries and suggest domain-specific specification and proof strategies.

3.1.1. Randomized algorithms

C. Paulin in collaboration with Ph. Audebaud from ENS Lyon, proposed a method for modeling probabilistic programs in *Coq* [48]. The method is based on a monadic interpretation of probabilistic programs as probability measures. A large *Coq* library has been developed and made publicly available (see also Section 5.12).

3.1.2. Floating-point programs

Many industrial programs (weather forecasts, plane trajectories, simulations...) use floating-point computations, typically double precision floating-point numbers [96]. Even if each computation is as good as it can be (except for elementary functions like sine, or exponential), the final result may be very wrong with no warnings, or the program will produce unexpected behaviors (like division by zero). This is the reason why guarantees should be provided to the user. We mean to guarantee for example that, for all or part of the possible inputs, the result obtained is correct (or near enough) and that no exceptional behavior will occur [58].

A high level of guarantee is obtained by formal proofs in *Coq*. We maintain and develop large *Coq* libraries for floating-point arithmetic: core definitions, axiomatic and computational rounding operations, high-level properties. It provides a framework for developers to formally certify numerical applications.A new such library is described in Section 5.9.

3.1.3. Certification of tools

Certifying the result of tools for analysing programs is a good challenge in the domain of proofs of higher-order functional programs. We obtained several results concerning formal proofs in *Coq* corresponding to automated deduction. These results are described in Section 3.3.

We have also an on-going project for the modeling and proof of the correctness of a compiler for the Lustre synchronous language. Our goal is to show the feasability of the certification using formal proofs of the compiler used in the new version of Scade developed by Esterel Technologies.

A PhD thesis started in Sep. 2009 has for main objective the development of a certified version the Frama-C/Jessie/Why verification chain.

3.2. Proof of Imperative and Object-Oriented programs

Participants: Romain Bardou, François Bobot, Sylvie Boldo, Jean-Christophe Filliâtre, Johannes Kanig, Claude Marché, Tuyen Nguyen, Andrei Paskevich, Christine Paulin-Mohring, Asma Tafat, Wendi Urribarrí.

A foundation step of the project is the PhD thesis of Jean-Christophe Filliâtre [6] that proposes to establish soundness of a program with imperative features (assignments, while loops, but also exceptions and exception handlers) by means of a translation into an equivalent purely functional program with logical annotations. Such an annotated functional program is very well-suited to be expressed in *Coq*'s type theory, hence this approach allowed for the first time to prove imperative programs with *Coq* [74].

Following this thesis, a new tool called *Why* was developed. It takes as input an imperative program and a specification that this program is expected to fulfil. It produces on one hand a set of *verification conditions* (VCs): logical formulas which have to be proved in the *Coq* system ; and on the other hand a *Coq*-term which contains a functional translation of the imperative program and a proof of correctness of this program based on the VCs. It was early remarked that this tool was independent of *Coq*, because the VCs can be validated in other interactive tools or with automatic provers. This multi-prover architecture is a powerful feature of *Why*: it spreads this technology well beyond the *Coq* community.

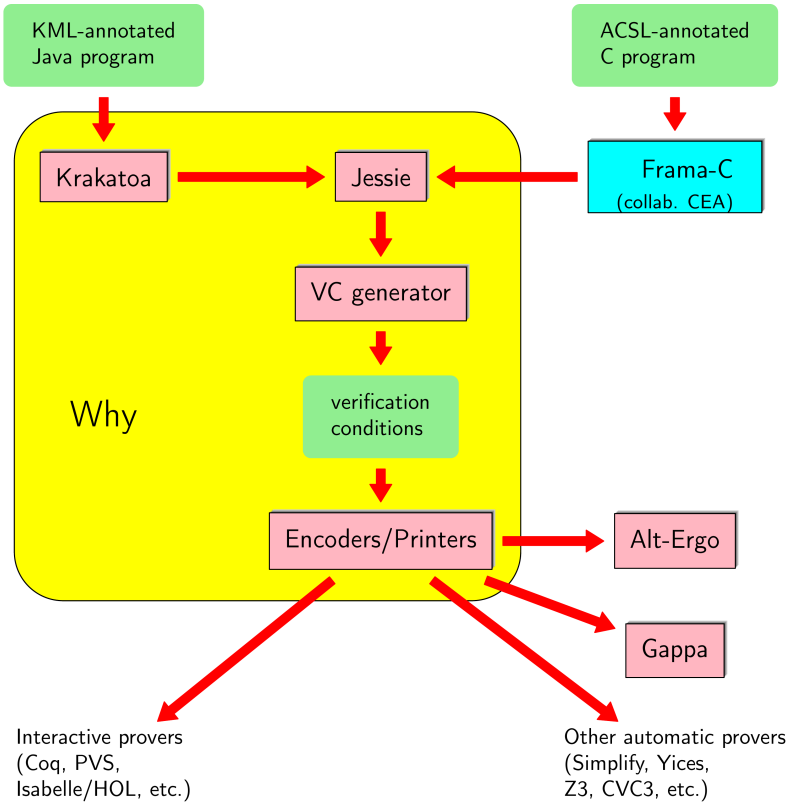

Figure 1. Architecture of certification chains: Frama-C, Why and back-end provers

3.2.1. The Why platform

Since 2002, we tackle programs written in mainstream programming languages. We first considered Java source code annotated with JML (Java Modeling Language). This method was implemented in a new tool called *Krakatoa* [10]. The approach is based on a translation from annotated Java programs into the specific language of *Why*, we then can reuse *Why*'s VCG mechanism and choose between different provers for establishing these VCs. From 2003, we followed the same approach for programs written in ANSI C [7].

The combination of the *Why* VC generator and the front-ends dealing with C or Java form a tool box for program verification, called the *Why* platform. Its overall architecture is shown on Figure 1. Nowadays, the front-end for C is in fact integrated in the Frama-C environment for static analysis of C programs (http://www.frama-c.cea.fr/), which was developed by the CEA-List in collaboration with us. Frama-C has an open architecture, structured as plugins around a shared kernel, and deductive verification of C code can be done using Why via the Jessie plugin. The annotation language for C source is also designed in collaboration with CEA, and called ACSL [52].

The central issue for the design of our platform is the modeling of memory heap for Java and C programs, handling possible aliasing (two different pointer or object expressions representing the same memory location): the *Why* VC generator does not handle aliasing by itself, indeed it does not support any form of complex data structures like objects, structures, pointers. On the other hand, it supports declaration of a kind of algebraic specifications: abstract data types specified by first-order functions, predicates and axioms. As a consequence, there is a general approach for using *Why* as a target language for *programming the semantics* of higher-level programming languages [90]. The *Krakatoa* and the Jessie memory models are inspired by the 'component-as-array' representation due to Bornat, following an old idea from Burstall, and commonly used to verify pointers programs [59]. Each field declaration f in a Java class or a C structure introduces a *Why* variable M_f in the model, which is a map (or an array) indexed by addresses. We extended this idea to handle Java arrays and JML annotations [10] and pointer arithmetic in C [7].

An important difficulty with programs handling pointers is to specify side-effects of a function or a method. The annotation languages offer the *assigns* clauses in specifications in order to delimitate the part of memory which is modified by a function or a method. We proposed an original modeling for such clauses [87] [7].

This kind of memory model does not scale up well for large programs. We designed an improved modeling of memory heap incorporating ideas from static analysis of memory separation, and from Reynolds' separation logic. Experiments on a C code proposed by Dassault Aviation were succesful [80], [79]

The use of *Why* as intermediate language opens interesting new approaches for reasoning on programs. We studied the specification of global properties, by reuse of the validation term of *Why* in order to define a model of each function, and then express and prove properties of functions composition. Such an approach was investigated by J. Andronick in the framework of proofs of security properties on smart cards [46], [45]. We also proposed a way to handle the Java Card transaction mechanism (a specificity of Java Card memory with both persistent and volatile parts), by indeed generating a *Why* model *on-the-fly* for each Java Card applet [88], thanks again to the flexibility of the approach using *Why* as an intermediate language.

3.2.2. Applications and case studies

The techniques we are developing can be naturally applied in domains which require to develop critical software for which there is a high need of certification.

The *Krakatoa* tool was successfully used for the formal verification of a commercial smart card applet [81] proposed by Gemalto. This case study have been conducted in collaboration with LOOP and Jive groups. Banking applications are concerned with security problems that can be the confidentiality and protection of datas, authentication, etc. The translation of such specifications into assertions in the source code of the program is an essential problem. We have been working on a Java Card applet for an electronic purse Demoney [62] developed by the company Trusted Logic for experimental purpose. Other Java Card case studies have been conducted in collaboration with Gemalto by J. Andronick and N. Rousset, in particular on global properties and Java Card transactions [46], [88].

To illustrate the effectiveness of the approach on C programs, T. Hubert and C. Marché performed a full verification of a C implementation of the Schorr-Waite algorithm [8], using *Coq* for the proofs. This is an allocation-free graph-marking algorithm used in garbage collectors, which is considered as a benchmark for verification tools. Other industrial case studies have been investigated by T. Hubert (with Dassault Aviation) [79] and by Y. Moy (with France Telecom) [91],[16].

3.3. Automated deduction

Participants: Sylvain Conchon, Évelyne Contejean, Stéphane Lescuyer, Claude Marché, Andrei Paskevich, Xavier Urbain.

Our group has a long tradition of research on automated reasoning, in particular on equational logic, rewriting, and constraint solving. The main topics that have been under study in recent years are termination proofs techniques, the issue of combination of decision procedures, and generation of proof traces. Our theoretical results are mainly materialized inside our two automated provers CiME and *Alt-Ergo*.

3.3.1. Termination

On the termination topic, we have studied new techniques which can be automated. A fundamental result of ours is a criterion for checking termination *modularly* and *incrementally* [98], and further generalizations [89]. These criteria and methods have been implemented into the CiME2 rewrite toolbox [5]. Around 2002, several projects of development of termination tools arose in the world. We believe we have been pioneer in this growth, and indeed we organized in 2004 the first competition of such tools.

A direction of research on termination techniques was also to apply our new approaches (for rewriting) to other computing formalisms, first to Prolog programs [92] and then to membership equational programs [73], a paradigm used in the *Maude* system [44].

3.3.2. Decision Procedures

3.3.2.1. Combination

Our research related to combination of decision procedures was initiated by a result [75] obtained in collaboration with Shankar's group at SRI-international who develops the PVS environment, showing how decision procedures for disjoint theories can be combined as soon as each of them provides a so-called "canonizer" and a "solver". Existing combination methods in the literature are generally not very well understood, and S. Conchon had a major contribution, in collaboration with Sava Krstić from OGI School of Science and Engineering (Oregon Health and Science University, USA), which is a uniform description of combination of decision procedures, by means of a system of inference rules, clearly distinguished from their strategy of application, allowing much clearer proofs of soundness and completeness [9], [69].

3.3.2.2. Polymorphic Logics

In the specific domain of program verification, the goals to be proved are given as formulae in a polymorphic multi-sorted first-order logic. Some of the sorts, such as integers and arrays, are built-in as they come from the usual data-types of programming languages. Polymorphism is used as a convenience for defining the memory models of C and Java programs and is handled at the level of the *Why* tool.

In order to be able to use all the available automated theorem provers (Simplify, SMT provers), including those which handle only untyped formulae (Simplify), one has to provide a way to get rid of polymorphism.

S. Conchon and É. Contejean have proposed an encoding of polymorphic multi-sorted logic (PSL) into unsorted logic based on term transformation, rather than addition of sort predicates which was used till then [72].

3.3.2.3. The Alt-Ergo theorem prover

It would be more convenient to deal with polymorphism directly in the theorem prover. There was no such prover available at the beginning of 2006, that is why S. Conchon and É. Contejean decided to develop a new tool called *Alt-Ergo* which is dedicated to the resolution of polymorphic and multi-sorted proof obligations and takes as input the *Why* syntax. In 2009, *Alt-Ergo* is still the only existing prover dealing with parametric polymorphism.

Alt-Ergo is based on $CC(X)$, a generic congruence closure algorithm developed in the team, for deciding ground formulas in the combination of the theory of equality with uninterpreted symbols and an arbitrary built-in solvable theory X. Currently, $CC(X)$ can be instantiated by the empty equational theory, by the linear arithmetics and the theory of constructors.

Alt-Ergo contains also a Fourier-Motzkin decision procedure for linear arithmetics inequalities, a home-made SAT-solver and an instantiation mechanism.

Alt-Ergo is safe and its architecture is modular: each part is described by a small set of inference rules and is implemented as an Ocaml functor. Moreover, the code is short (6500 lines).

3.3.3. Automated proofs and certificates

A common issue to both termination techniques and decision procedures is that automatic provers use complex algorithms for checking validity of formula or termination of a computation, but when they answer that the problem is solved, they do not give any more useful information. It is highly desirable that they give a *proof trace*, that is some kind of certificate that could be double-checked by a third party, such as an interactive proof assistant like *Coq*. Indeed *Coq* is based on a relatively small and stable kernel, so that when it checks that a proof is valid, it can be trusted. Morevoer, a subpart of Coq has been proven correct in Coq [50].

3.3.3.1. Coccinelle and CiME's traces

In addition to efficient termination techniques, CiME implements in particular a semi-decision procedure for the equality modulo a set of axioms, based on ordered completion. In 2005, the former human readable proof traces have been replaced by *Coq* certificates, based on reified proof objects for a FOL logic modelled inside *Coq* [70].

É. Contejean, A. Paskevich, X. Urbain and the Cédric participants of the A3PAT project, Pierre Courtieu, Olivier Pons (CNAM), and Julien Forest, (ENSIIE) develop the new version of the CiME tool, CiME 3, associated with a *Coq* library called Coccinelle developed by É. Contejean. A trace generator outputs a trace for *Coq* in the unified framework provided by the Coccinelle library [71][4]. Coccinelle contains the corresponding modelling of terms algebras and rewriting statements, and also some generic theorems which are needed for establishing a rewriting property from a trace. For example, in order to produce a certificate of termination for a rewriting system, one may provide as a trace an ordering that contains the rewrite system, but it is also needed to have a proof that this ordering is well-founded. Such a proof (for RPO for instance) is part of Coccinelle as a generic property. Coccinelle also contains as generic theorems some powerful criteria of termination: dependency pairs [47], the main modularity theorem for termination presented in the thesis of Urbain [98] as well as innermost termination, dependency pairs for it and its equivalence with standard termination in some specific cases [77].

The main improvement over the previous approach [70] is that the *Coq* development is parameterized with respect to the equality predicate (instead of using the *Coq* native equality). This allows to deal uniformly with equality modulo a set of axioms, with termination of a set of rewrite rules, and with rewriting modulo a set of equations, such as associativity-commutativity.

Certifying termination proofs gained interest in the term rewriting community. Groups are either developing their own certifier, or producing traces for other's, thanks to a shared XML format. Since 2007, the termination competition has a category for certified termination proofs.

3.4. Synchronous Programming

Participants: Cédric Auger, Léonard Gérard, Louis Mandel, Florence Plateau, Marc Pouzet.

The goal is to propose high-level languages for the development of critical embedded systems with both high temporal requirements and safety [53], [78], [54], [60]. Our research activities concern the extension of synchronous languages with richer abstraction mechanisms (e.g., higher-order, functionality, dedicated type systems such as the clock calculus), the ability to describe heterogeneous systems (e.g., data-flow and control-flow, discrete and continous) or to account for resources through dedicated type-systems.

These research activities are experimented inside two programming languages, Lucid Synchrone and ReactiveML.

Lucid Synchrone is a data-flow language based on a Lustre semantics and is dedicated to real-time embedded software. It extends Lustre with features usually found in ML-languages such as typing and higher-order functions. It provides original features such as the arbitrary mix of data-flow and hierarchical automata [3] [66], various type-based static analysis [67], [68] and modular compilation into sequential code [61], [57] [56].

ReactiveML [86] is an extension of Objective Caml with synchronous concurrency (based on synchronous parallel composition and broadcast of signals). The goal is to provide a general model of deterministic concurrency inside a general purpose functional language to program reactive systems (e.g., graphical interfaces, simulation systems). The research activity concerns the development of compilation techniques, dedicated type systems to ensure various safety properties (e.g., determinism, reactivity, boundedness) or the mix of both synchronous and asynchronous concurrency.

3.4.1. Extended synchronous models

In collaboration with Albert Cohen and Christine Eisenbeis (INRIA Alchemy), Marc Duranton (Philips Natlabs, Eindhoven), we have introduced in 2005 a new programming model for the design of video intensive applications. This model, called the n-synchronous model, is based on an extension of the synchronous model allowing to combine non strictly synchronous streams provided that they can be synchronized through the use of bounded buffers. This is obtained by introducing particular clocks as infinite binary periodic words [99]. Thanks to the periodic nature of these clocks, we are able to verify properties like the absence of buffer overflows and deadlocks during the execution. Clock verification is expressed as a type-inference problem with a sub-typing rule. The core of the model has been settled in [63] and [64]. We introduced a notion of abstractions for these clocks as a mean to reason about sets of (non necessarily periodic) clocks [65]. We defined the programming language Lucy-n to program with the n-synchronous model [33], [27].

3.4.2. The semantics of hybrid system modelers

Hybrid systems modelers have become the corner stone of embedded system development, with Simulink a de facto standard and Modelica a new player. They allow both *discrete* controllers and their *continuous* environments to be expressed *in a single language*. Despite the availability of such tools, there remain a number of issues related to the lack of reproducibility of simulations and to the separation of the continuous part, which has to be exercised by a numerical solver, from the discrete part, which must be guaranteed not to evolve during a step. Such tools still raise a number of issues that, we believe, require more fundamental understanding.

4. Application Domains

4.1. Panorama

Many systems in telecommunication, banking or transportation involve sophisticated software for controlling critical operations. One major problem is to get a high-level of confidence in the algorithms or protocols that have been developed inside the companies or by partners.

Many smartcards in mobile phones are based on a (small) Java virtual machine. The card is supposed to execute applets that are loaded dynamically. The operating system itself is written in C, it implements security functions in order to preserve the integrity of data on the card or to offer authentication mechanisms. Applets are developed in Java, compiled, and then the byte-code is loaded and executed on the card. Applets or the operating systems are relatively small programs but they need to behave correctly and to be certified by an independent entity.

If the user expresses the expected behavior of the program as a formal specification, it is possible for a tool to check whether the program actually behaves according to the requirements. We have a collaboration with Gemalto in this area.

Avionics or more generally transportation systems are another area were there are critical algorithms involved, for instance in Air Traffic control. We have collaborations in this domain with Dassault-Aviation and National Institute of Aerospace (NIA, Hampton, USA).

5. Software

5.1. The CiME rewrite toolbox

Participants: Évelyne Contejean [contact], Claude Marché, Andrei Paskevich, Xavier Urbain.

CiME is a rewriting toolbox. Distributed since 1996 as open source, at URL http://cime.lri.fr. Beyond a few dozens of users, CiME is used as back-end for other tools such as the TALP tool developed by Enno Ohlebusch at Bielefeld university for termination of logic programs; the MU-TERM tool (http://www.dsic.upv.es/~slucas/csr/termination/muterm/) for termination of context-sensitive rewriting; the CARIBOO tool (developed at INRIA Nancy Grand-Est) for termination of rewriting under strategies; and the MTT tool (http://www.lcc.uma.es/~duran/MTT/) for termination of Maude programs. CiME2 is no longer maintained, and the currently developed version is CiME3, available at http://a3pat.ensiie.fr/pub. The main new feature of CiME3 is the production of traces for *Coq*. CiME3 is also developed by the participants of the A3PAT project at the CNAM, and is distributed under the Cecill-C licence.

5.2. The Why platform

Participants: Jean-Christophe Filliâtre [contact], Romain Bardou, François Bobot, Claude Marché, Guillaume Melquiond, Andrei Paskevich.

The *Why* platform is a set of tools for deductive verification of Java and C source code. In both cases, the requirements are specified as annotations in the source, in a special style of comments. For Java (and Java Card), these specifications are given in JML and are interpreted by the *Krakatoa* tool. For C, we designed our own specification language, largely inspired from JML. Those are interpreted by the *Caduceus* tool.

The platform is distributed as open source, under GPL Licence, at http://why.lri.fr/.

A back-end tool also called *Why* serves as the VCG. It differs from other systems in that it outputs conditions for several existing provers: interactive ones (*Coq*, Isabelle/HOL, PVS, HOL-light, Mizar) and automatic ones (Simplify, *Alt-Ergo*, Gappa, and SMT provers Yices, CVC3, Z3, haRVey, etc.). The *Why* VCG alone has been used by external researchers in published verifications of non-trivial algorithms (Efficient square root used in GMP [55], Knuth's algorithm for prime numbers [97]).

Krakatoa is used internally at Gemalto company for security analyses. *Krakatoa* is also used for teaching (University of Evry, Ecole Polytechnique).

Caduceus was experimented at Gemalto company, at Dassault Aviation company, and at CEA (Saclay). It was also used for teaching at Ecole Polytechnique (2006/2007, 1st year master ISIC, *projet de verification*) and at University of Evry (2005-2006 and 2006-2007, proofs using *Coq*). It is now subsumed by the Jessie plugin of *Frama-C*. The latter is under use by several groups in the world, e.g at Fraunhofer Institute in Berlin [76], and at Universidade do Minho in Portugal [20].

5.3. The Alt-Ergo theorem prover

Participants: Sylvain Conchon [contact], Évelyne Contejean, Stéphane Lescuyer, Alain Mebsout, Mohamed Iguernelala.

Alt-Ergo is an automatic, little engine of proof dedicated to program verification, whose development started in 2006. It is fully integrated in the program verification tool chain developed in our team. It solves goals that are directly written in the *Why*'s annotation language; this means that *Alt-Ergo* fully supports first order polymorphic logic with quantifiers. *Alt-Ergo* also supports the standard [93] defined by the SMT-lib initiative.

It is currently used in our team to prove correctness of C and Java programs as part of the *Why* platform. *Alt-Ergo* is also called as an external prover by the Pangolin tool developed by Y. Regis Gianas, INRIA project-team Gallium http://code.google.com/p/pangolin-programming-language/. Alt-Ergo is usable as a back-end prover in the SPARK verifier for ADA programs, since Oct 2010. It is planed to be integrated in next generation of Airbus development process.

Alt-Ergo is distributed as open source, under the CeCILL-C licence, at URL http://alt-ergo.lri.fr.

5.4. Lucid Synchrone

Participant: Marc Pouzet [contact].

Lucid Synchrone is an experimental language for the implementation of reactive systems. It is based on the synchronous model of time as provided by Lustre combined with features from ML languages. It provides powerful extensions such as type and clock inference, type-based causality and initialization analysis and allows to arbitrarily mix data-flow systems and hierarchical automata or flows and valued signals.

It is distributed under binary form, at URL http://www.lri.fr/~pouzet/lucid-synchrone/.

The language has served as a laboratory to experiment various extensions of the language Lustre. Several programming constructs (e.g. merge, last) and type-based program analysis (e.g., typing, clock calculus) originaly introduced in Lucid Synchrone are integrated in the new SCADE 6 compiler developped at Esterel-Technologies.

5.5. Reactive ML

Participant: Louis Mandel [contact].

ReactiveML is a programming language dedicated to the implementation of interactive systems as found in graphical user interfaces, video games or simulation problems. ReactiveML is based on the synchronous reactive model due to Boussinot, embedded in an ML language (Objective Caml).

The Synchronous reactive model provides synchronous parallel composition and dynamic features like the dynamic creation of processes. In ReactiveML, the reactive model is integrated at the language level (not as a library) which leads to a safer and a more natural programming paradigm.

ReactiveML is distributed at URL http://www.lri.fr/~mandel/rml. The compiler is distributed under the terms of the Q Public License and the library is distributed under the terms of the GNU Library General Public License. The development of ReactiveML started at the University Paris 6 (from 2002 to 2006).

The language is mainly used for the simulation of mobile ad hoc networks at the University Paris 6 and for the simulation of sensor networks at France Telecom and Verimag (CNRS, Grenoble).

5.6. Bibtex2html

Participants: Jean-Christophe Filliâtre [contact], Claude Marché.

Bibtex2html is a generator of HTML pages of bibliographic references. Distributed as open source since 1997, under the GPL licence, at http://www.lri.fr/~filliatr/bibtex2html/. We estimate that between 10000 and 100000 web pages have been generated using Bibtex2html.

Bibtex2html is also distributed as a package in most Linux distributions. Package popularity contests show that it is among the 20% most often installed packages.

5.7. Ocamlgraph

Participants: Jean-Christophe Filliâtre [contact], Sylvain Conchon.

Ocamlgraph is a graph library for Objective Caml. It features many graph data structures, together with many graph algorithms. Data structures and algorithms are provided independently of each other, thanks to Ocaml module system. Ocamlgraph is distributed as open source, under the LGPL licence, at http://ocamlgraph.lri.fr/. It is also distributed as a package in several Linux distributions. Ocamlgraph is now widely spread among the community of Ocaml developers.

5.8. Mlpost

Participants: Jean-Christophe Filliâtre [contact], Johannes Kanig, Stéphane Lescuyer, Romain Bardou, François Bobot.

Mlpost is a tool to draw scientific figures to be integrated in LaTeX documents. Contrary to other tools such as TikZ or MetaPost, it does not introduce a new programming language; it is instead designed as a library of an existing programming language, namely Objective Caml. Yet it is based on MetaPost internally and thus provides high-quality PostScript figures and powerful features such as intersection points or clipping. Mlpost is distributed as open source, under the LGPL licence, at http://mlpost.lri.fr/. Mlpost was presented at JFLA'09 [49].

5.9. The Flocq library

Participants: Sylvie Boldo [contact], Guillaume Melquiond.

The Flocq library for the *Coq* proof assistant is a comprehensive formalization of floating-point arithmetic: core definitions, axiomatic and computational rounding operations, high-level properties. It provides a framework for developers to formally certify numerical applications.

It is distributed as open source, under a LGPL license, at http://flocq.gforge.inria.fr/. It was first released in 2010.

5.10. The Gappa tool

Participant: Guillaume Melquiond [contact].

Given a logical property involving interval enclosures of mathematical expressions, Gappa tries to verify this property and generates a formal proof of its validity. This formal proof can be machine-checked by an independent tool like the *Coq* proof-checker, so as to reach a high level of confidence in the certification [15], [17].

Since these mathematical expressions can contain rounding operators in addition to usual arithmetic operators, Gappa is especially well suited to prove properties that arise when certifying a numerical application, be it floating-point or fixed-point. Gappa makes it easy to compute ranges of variables and bounds on absolute or relative roundoff errors.

Gappa is being used to certify parts of the mathematical libraries of several projects, including CRlibm, FLIP, and CGAL. It is distributed as open source, under a Cecill-B / GPL dual-license, at http://gappa.gforge.inria.fr/. Part of the work on this tool was done while in the Arénaire team (INRIA Rhône-Alpes), until 2008.

5.11. The Interval package for Coq

Participant: Guillaume Melquiond [contact].

The Interval package provides several tactics for helping a *Coq* user to prove theorems on enclosures of real-valued expressions. The proofs are performed by an interval kernel which relies on a computable formalization of floating-point arithmetic in *Coq*.

It is distributed as open source, under a LGPL license, at http://www.lri.fr/~melquion/soft/coq-interval/. Part of the work on this library was done while in the Mathematical Components team (Microsoft Research–INRIA Joint Research Center).

In 2010, the Flocq library was used to straighten and fill the floating-point proofs of the Interval package.

5.12. The Alea library for randomized algorithms

Participants: Christine Paulin-Mohring [contact], David Baelde.

The ALEA library is a *Coq* development for modeling randomized functional programs as distributions using a monadic transformation. It contains an axiomatisation of the real interval $[0, 1]$, a definition of distributions and general rules for approximating the probability that a program satisfies a given property.

It is distributed as open source, at http://www.lri.fr/~paulin/ALEA. It is currenltly used as a basis of the Certicrypt environment (MSR-INRIA joint research center, Imdea Madrid, INRIA Sophia-Antipolis) for formal proofs for computational cryptography [51].

5.13. The Coccinelle library for term rewriting

Participant: Évelyne Contejean [contact].

Coccinelle is a Coq library for term rewriting. Besides the usual definitions and theorems of term algebras, term rewriting and term ordering, it also models some of the algorithms implemented in the CiME toolbox, such a matching, matching modulo associativity-commutativity, computation of the one-step reducts of a term, RPO comparison between two terms, etc. The RPO algorithm can effectively be run inside Coq, and is used in the Color developpement (http://color.inria.fr/) as well as for certifying Spike implicite induction theorems in Coq (Sorin Stratulat).

Coccinelle is developped by Évelyne Contejean, available at (http://www.lri.fr/~contejea/Coccinelle), and is distributed under the Cecill-C licence.

6. New Results

6.1. Floating-Point Programs

- A. Ayad and C. Marché presented at IJCAR [19] the consolidated model of floating-point computations, as it is now implemented in the Jessie plugin of Frama-C. It takes into account the full IEEE 754 standard: special values for infinities and not-a-number, different rounding modes, etc. It also supports several provers: automatic proofs with SMT solvers, automatic proofs on rounding errors and overflow with Gappa, and interactive proofs with Coq for hard-to-prove obligations.
- S. Boldo and J.-M. Muller (CNRS, Arénaire, LIP, ÉNS Lyon) have worked on new floating-point algorithms for computing the exact and approximated errors of the FMA (fused multiply-and-add). This article is accepted in IEEE Transactions on Computers [14].
- S. Boldo and T. Nguyen have worked on how to prove numerical programs on multiple architectures [23]. T. Nguyen presented her poster [43] at the Digiteo Forum on October 12th 2010.
- S. Boldo, F. Clément, J.-C. Filliâtre, M. Mayero, G. Melquiond and P. Weis have worked on the formal proof of the method error of a numerical scheme [22]. This is to be joined with the corresponding rounding error that was also formally proved [1].
- S. Boldo, as invited speaker at the NSV workshop, published an article about the formal verification of numerical programs in Coq [18].
- G. Melquiond participated to a handbook on floating-point arithmetic (IEEE-754r standard, hardware and software implementations, programming languages, and so on) coordinated by J.-M. Muller [36].

- G. Melquiond, in collaboration with F. de Dinechin (Arénaire, LIP, ÉNS Lyon) and C. Lauter (Intel Hillsboro), improved the methodology for formally proving floating-point mathematical functions when their correctness depends on relative errors [17].
- Collaboration between the Gappa, *Coq*, and Frama-C/Jessie tools was improved by making the Gappa support library depend on the Flocq library for the core formalization of floating-point arithmetic.

6.2. Models and Proofs of Imperative Programs

- A new major version of the *Why* platform started in 2010. The main developers are A. Paskevich, J.-C. Filliâtre and F. Bobot. The language of *Why*, both programming and annotation parts, was significantly extended [39]: algebraic types, records, pattern matching, recursive logical definitions are now supported. These logical declarations are structured in modules (a.k.a. theories). The module language comes with an original mechanism for reusing theories in specialized contexts using partial instantiations.

  The main efforts are put into creation of a well-designed modular programming interface, allowing to extend *Why* easily with new modules (parsers, proof task transformations, prover interfaces) as well as to embed *Why* into third-party tools. A large effort was also invested into the design of proof task transformations and encodings used to send goals to external provers. In particular, S. Cruanes has implemented a new type-elimination encoding and a pretty-printer targeted at the provers of the TPTP family. A first release of the new version of *Why* is planned for December 2010.
- A. Tafat and C. Marché, together with S. Boulmé from VERIMAG, Grenoble, proposed a new approach based on data refinement techniques for building certified object-oriented program in a modular way. A first version appeared as an INRIA research report [40]. An improved version was presented at the international conference FoVeOOS, and appeared in its proceedings [28].
- Specification of functional behavior of generic Java programs, typically those of the Standard Java API, are difficult to design in a reusable way. C. Marché, together with E. Tushkanova, A. Giorgetti and O. Kouchnarenko from LIFC, Besançon, proposed extensions of the standard behavioral specification languages (like JML or ACSL) to support generic types but also a kind of higher-order setting via instantiation of interfaces parameterized by theories. This was presented at the LDTA workshop [29], satellite of the joint ETAPS conferences.
- Proving that pointer programs preserve data invariants, in a modular way, is known to be a challenge. R. Bardou and C. Marché are designing a new approach based on advanced static typing systems (based on memory regions and permissions) to control memory updates and ownership of data. A preliminary description of the resulting language is described in an INRIA research report [38]. A prototype implementation is built, called Capucine. An initial version has been available for download at http://romain.bardou.fr/capucine. To demonstrate the ability of this approach, one of the programs of the VACID-0 benchmarks [82] as been certified. This case study is going to be presented at the JFLA conference [30].
- Another challenge is the treatment of imperative features in the context of higher-order programming languages. J. Kanig has developed a new approach, based on a Hoare logic for higher-order systems and a higher-order effect system, that deals with these features in a modular way. A prototype called Who has been developed and is available for download at http://www.lri.fr/~kanig/who.html. A number of case studies, mixing imperative and higher-order features have been realized using this tool. J. Kanig has defended his PhD thesis [11] on November 26th, 2010.
- Barbará Vieira (Ph.D. student at Universidade do Minho, Braga, Portugal), visited ProVal from March to May 2009. In collaboration with J.-C. Filliâtre, she developed a tool for the verification of CAO programs. CAO is a domain-specific language for cryptographic protocols. This work was presented at OpenCert 2010 [20].

- Wendi Urribarrí as part of her PhD thesis developed a module calculus for the *Why* programming language. She designed a refinement calculus which allows to implement a module interface possibly changing the representation of private data. For instance a module implementing finite sets will be specified using an mathematical notion of sets that will be made available to possible clients of the module and implemented using efficient data structures like arrays or balanced trees. She obtained a logical criteria under which two modules working on shared data can be simultaneously loaded without breaking the local invariants. A prototype implementation has been developed and will be used for experimenting with significant examples.

6.3. Automatic Generation of Specifications

- A long article presenting the work of Y. Moy and C. Marché on the automatic generation of program annotations using abstract interpretation appeared in the Journal of Symbolic Computation [16].
- K. Krishnamani presented a work on predicate abstraction integrating binary decision diagrams and SMT solvers, at the DATE conference [24] for verification of hybrid systems. This approach is implemented in the tool NuSMT, available at http://www.lri.fr/~kalyan/nusmt/. Together with C. Marché, they are applying predicate abstraction techniques to discover program invariants. A new Frama-C plug-in will use the above technique to automatically insert ACSL annotations in the C source code.

6.4. Certification

- P. Herms formalized in Coq a verification condition generator for a core language close to Why. It is formally proved that the validity of VCs guaranttes that the program respects its specifications. A certified OCaml program can be produced using the Coq extraction mechanism. An early version of this work was presented at the Coq Workshop [35], satellite of the joint Floc 2010 conference. Writing a complete article is under progress.
- S. Lescuyer has formalized in Coq the algorithms at the heart of the Alt-Ergo SMT solver. This formalization represents a propositional SAT solver combined with an decision procedure for equality modulo linear arithmetic over integers. Using the technique of reflection, the development has been turned into a tactic which can be used in Coq to automatically discharge goals in this logical fragment.
- A version of CiME 3 is released. The new certification and proof engines include the new termination criterion we developed, and that was presented at the PEPM symposium in January [25] together with new formalization techniques for graphs in Coq. CiME 3 is to date the only tool able to prove and certify confluence as well as termination of term rewriting systems.

6.5. Contributions to functional programming environments

- In collaboration with F. Le Fessant (INRIA Saclay-Île-de-France), S. Conchon and J.-C. Filliâtre supervised in 2009 the summer project of G. Von Tokarski and J. Robert (graduate students from Université Paris-Sud), funded by Jane Street Capital (NYC, USA). Together they designed and implemented Ocamlviz, a tool for real-time monitoring of Objective Caml programs. Ocamlviz is available at http://ocamlviz.forge.ocamlcore.org/. This work was presented at the JFLA conference [31].
- J.-C. Filliâtre and K. Krishnamani designed a distributed computing library for Objective Caml which facilitates distributed execution of parallelizable computations in a seamless fashion. It is re-usable thanks to its polymorphic API, and incorporates a robust fault-tolerant mechanism. This is published at the JFLA conference [32] and available at http://functory.lri.fr/. It is being put to use for computationally intensive verification tasks within our team and the results are encouraging.

6.6. Synchronous Programming

- F. Plateau defended her PhD. thesis in january 2010 on the *n*-synchronous model.
- The paper *Modular Static Scheduling of Synchronous Data-flow Programs* by M. Pouzet and P. Raymond (VERIMAG Grenoble) has been selected among the two best papers at EMSOFT 2009. An extended version is published in a special issue of the *Journal of Design Automation for Embedded Systems*, in 2010.
- M. Pouzet will present a work done with Albert Benveniste and Benoit Caillaud (INRIA Rennes) at the *49th Conference on Design and Control (CDC)* on the use of non-standard analysis as a semantics basis for hybrid system modelers *a la Simulink*.

6.6.1. The n-synchronous model

The n-synchronous model introduced a way to compose streams which have *almost the same clock* and can be synchronized through the use of a finite buffer. We have designed the language Lucy-n to program in this model of computation [34], [27]. This language is similar to the first order synchronous data-flow language Lustre in which a buffer operator is added. A dedicated type system allows to check that programs can be executed in bounded memory and to compute the buffers sizes needed. Technically it is done through the introduction of a subtyping constraint at each bufferization point. To solve the subtyping constraints we have defined an algorithm that uses an improved version of the abstraction introduced in [65]. We have proved the correctness properties of this new abstraction in Coq.

We also worked on new typing algorithms that do not use clock abstraction and thus allows to model Latency Insensitive Designin Lucy-n [26].

6.6.2. The semantics of hybrid system modelers

In collaboration with Albert Benveniste and Benoit Caillaud (INRIA Rennes), M. Pouzet have proposed using non standard analysis as a semantic domain for hybrid systems. Non standard analysis is an extension of classical analysis in which infinitesimals can be manipulated as first class citizens. This allows us to provide a denotational semantics and a constructive semantics for hybrid systems, thus establishing simulation engines on a firm mathematical basis. In passing, we cleanly separate the job of the numerical analyst (solving differential equations) from that of the computer scientist (generating execution schemes). This work will appear in the proceeding of the *49th Conference on Design and Control* in 2010 [21].

The other part of this work concern static typing and compilation. Starting from a minimal, yet full-featured, Lustre-like synchronous language, we have proposed a conservative extension where data-flow equations can be mixed with ordinary differential equations (ODEs) with possible reset. A type system is proposed to statically distinguish discrete computations from continuous ones and to ensure that signals are used in their proper domains.

The extended data-flow language is realized through a source-to-source transformation into a synchronous subset, which can then be compiled using existing tools into routines that are both efficient and bounded in their use of memory. These routines are orchestrated with a single off-the-shelf numerical solver using a simple but precise algorithm which treats causally-related cascades of zero-crossings. We have validated the viability of the approach through experiments with the Syndials library.

6.6.3. Dynamic aspects in synchronous languages

We have continued the development of ReactiveML, an extension of Objective Caml with reactive constructs [86]. A lecture [33] presenting the language has been done during the JFLA 2010 conference.

We are working on the interactive definition of reactive systems as a kind of extreme dynamic aspect. It allows to write a program and add it to an application which is already running. The ReactiveML toplevel add these features to the ReactiveML language. A journal article about the toplevel has been submitted. In addition, we are working on a new scripting language based on the synchronous reactive model in collaboration with Frédéric Boussinot, Pejman Attar (INRIA Sophia) and Jean-Fredy Susini (CNAM).

7. Contracts and Grants with Industry

7.1. System@tic: Hi-Lite

Participants: Claude Marché [contact], Jean-Christophe Filliâtre, Sylvain Conchon, Evelyne Contejean, Andrei Paskevich, Alain Mebsout, Mohamed Iguernelala.

The Hi-Lite project (http://www.open-do.org/projects/hi-lite/) is a project in the SYSTEM@TIC Paris Region French cluster in complex systems design and management http://www.systematic-paris-region.org.

Hi-Lite is a project aiming at popularizing formal methods for the development of high-integrity software. It targets ease of adoption through a loose integration of formal proofs with testing and static analysis, that allows combining techniques around a common expression of specifications. Its technical focus is on modularity, that allows a divide-and-conquer approach to large software systems, as well as an early adoption by all programmers in the software life cycle.

Our involvements in that project include the use of the Alt-Ergo prover as back-end to already existing tools for SPARK/ADA, and the design of a verification chain for an extended SPARK/ADA language to verification conditions, via the Why VC generator.

This project is funded by the french ministry of industry (FUI), the Île-de-France region and the Essonne general council for 36 months from September 2010.

7.2. CEA-Airbus contract

Participants: Sylvain Conchon [contact], Évelyne Contejean, Claude Marché.

In conjunction with the INRIA funding of ADT Alt-Ergo, a specific support contract has started in Sep 09, between INRIA, CEA Saclay and Airbus France at Toulouse. This is to support our efforts for the maintainance and to feature updates of Alt-Ergo, for its use at Airbus software development and certification of avionics critical code.

7.3. Airbus contract

Participant: Sylvain Conchon [contact].

This support contract has started in Sep 10, between INRIA and Airbus France at Toulouse. This is to support our efforts for the DO-178B qualification of Alt-Ergo.

8. Other Grants and Activities

8.1. Regional Initiatives

8.1.1. Hisseo

Participants: Sylvie Boldo [contact], Claude Marché, Guillaume Melquiond, Thi-Minh-Tuyen Nguyen.

Hisseo is a 3 years Digiteo project that started in September 2008. http://hisseo.saclay.inria.fr

The Hisseo project focuses on the problems related to the treatment of floating-point computations in the compilation process, especially in the case of the compilation of critical C code.

Partners: CEA List (Saclay), INRIA Paris-Rocquencourt (Team Gallium).

8.1.2. Pactole

Participants: Évelyne Contejean, Jean-Christophe Filliâtre, Xavier Urbain [contact].

Pactole is a 3 year Digiteo project which started in October 2009.

The Pactole project focuses on automation and formal verification for ubiquitous, large scale environments. Tasks include proof automation techniques for distributed systems, verification conditions for fault tolerant distributed systems, specification and design of fundamental services for mobile sensor networks. The principal investigator of Pactole is Xavier Urbain.

Partners: CÉDRIC (CNAM/ENSIIE), LIP6 (UPMC).

8.2. National initiatives

8.2.1. U3CAT

Participants: Jean-Christophe Filliâtre, Claude Marché [contact], Guillaume Melquiond, Kalyan Krishnamani, Asma Tafat, Paolo Herms.

U3CAT (Unification of Critical C Code Analysis Techniques) is a project funded by ANR within its programme "Systèmes Embarqués et Grandes Infrastructures - ARPEGE". It aims at verification techniques of C programs, and is partly a follow-up of the former CAT project. It started in January 2009 and will end in 2012.

The main goal of the project is to integrate various analysis techniques in a single framework, and make them cooperate in a sound way. We address the following general issues:

- Verification techniques for floating-point programs;
- Specification and verification of dynamic or temporal properties;
- Combination of static analysis techniques;
- Management of verification sessions and activities;
- Certification of the tools chains for compilation and for verification.

Partners: CEA-List (Saclay, project leader), Lande team (INRIA Rennes), Gallium team (INRIA Rocquencourt), Dassault Aviation (Saint-Cloud), Airbus France (Toulouse), ATOS Origin (Toulouse), CNAM Cedric laboratory (Evry), CS Communication & Systems (Toulouse), Hispano-Suiza/Safran (Moissy-Cramayel).

8.2.2. INRIA ADT Alt-Ergo

Participants: Sylvain Conchon [contact], Evelyne Contejean, Claude Marché, Alain Mebsout, Mohamed Iguernelala.

The *ADT* (Action de Développement Technologique) Alt-Ergo is a 2-years project funded by INRIA, started in September 2009.

The goal is the maturation of the Alt-Ergo prover towards its use in an industrial context in particular for avionics. The expected outcomes of this ADT are the following:

- improving the efficiency of Alt-Ergo;
- fine tuning of Alt-Ergo for the SMT competition;
- generation of counter-examples;
- the qualification of Alt-Ergo for the norm DO-178B.

External Collaborators: Airbus France (Toulouse), Dassault Aviation (Saint-Cloud), team Typical (INRIA, École Polytechnique).

8.2.3. FOST

Participants: Sylvie Boldo [contact], Jean-Christophe Filliâtre, Guillaume Melquiond.

FOST (Formal prOofs of Scientific compuTation programs) is a 3 years ANR "Blanc" project started in January 2009. S. Boldo is the principal investigator of this project. http://fost.saclay.inria.fr

The FOST project follows CerPAN's footprints as it aims at developing new methods to bound the global error of a numerical program. These methods will be very generic in order to prove a large range of numerical analysis programs. Moreover, FOST aims at providing reusable methods that are understandable by non-specialists of formal methods.

Partners: University Paris 13, INRIA Paris - Rocquencourt (Estime).

8.2.4. SIESTA

Participant: Marc Pouzet.

SIESTA is a 4 year project funded by ANR RNTL. The coordinator is Y. Parissis (LIG, Grenoble). http://www.siesta-project.com. The project started in january 2008.

This project addresses the automated testing of embedded systems implemented in SCADE or Simulink. M Pouzet is involved on the architecture of the SCADE 6 compiler to integrate verification techniques. The challenge is to take new programming constructs (e.g., hierarchical automata, reset and general forms of clocks) into account to improve verification techniques and modularity.

Partners: AIRBUS, Turbomea, Hispano-Suiza, Onera, CEA List, Esterel-Technologies, EADS, LRI, LIG.

8.2.5. GENCOD

Participant: Marc Pouzet.

This project is funded by DGE for two years. The coordinator is Thales. The project started in sept. 2008 (delayed by one year).

The GENCOD project aims at defining methods to certify the Esterel compiler for hardware (Norm. DO 254, the hardware version of DO 178 B used for critical software).

Partners: Dassault-Aviation, Thales, Esterel-Technologies, TNI, Airbus, LRI.

8.2.6. INRIA Action d'Envergure Synchronics

Participants: Marc Pouzet, Louis Mandel.

This project is funded by INRIA for 4 years and started in Jan. 2008. The coordinators are A. Girault (INRIA Rhône Alpes) and M. Pouzet. http://synchronics.inria.fr/

The goal of the project is to propose new languages for the development of embedded systems allowing *from a unique source* to both simulate the system with its environment and generate code. It capitalizes on recent extensions of data-flow synchronous languages (Lucid Synchrone, ReactiveML), a relaxed form of synchrony, and means to mix discrete and continous systems inside the synchronous model of time.

The project focuses on language extensions to increase modularity, dedicated type systems to ensure safety properties, efficient compilation and the mix of discrete and continuous time.

Partners: INRIA Rhône Alpes (Gwenaël Delaval, Alain Girault, Bertrand Jeannet), IRISA (Benoit Caillaud), VERIMAG (Erwan Jahier, Pascal Raymond), INRIA Saclay (Albert Cohen, Marc Pouzet, Louis Mandel)

8.2.7. SCALP

Participants: Christine Paulin-Mohring [contact], Xavier Urbain.

This project is funded by ANR (program SESUR). http://scalp.gforge.inria.fr/

It started on january 2008 for 4 years; the coordinator is Yassine Lakhnech from VERIMAG.

Partners: Verimag, INRIA Sophia-Antipolis(Everest then Marelle team), ENS Lyon, LRI, CNAM.

The SCALP project (Security of Cryptographic Algorithms with Probabilities) aims at developping automated tools for the verification of cryptographic systems.

8.2.8. PARTOUT

Participants: Louis Mandel, Marc Pouzet.

This project is funded by ANR (program DEFIS). http://www-sop.inria.fr/mimosa/PARTOUT

It started on january 2009 for 4 years; the coordinator is Frédéric Boussinot from INRIA Mimosa.

Partners: INRIA Mimosa, CNAM, LRI.

The goal of the project PARTOUT is, from a programming language point of view, to study the impact on programming of the globalization of parallelism which now covers all the spectrum of informatics, ranging from multicore architectures and distributed systems, up to applications deployed on the Web.

8.2.9. DECERT

Participants: Sylvain Conchon, Évelyne Contejean, Stéphane Lescuyer.

DECERT (DEduction and CERTification) is an ANR "Domaines Emergents" project. It started on January 2009 for 3 years; the coordinator is Thomas Jensen from the Lande team of IRISA/INRIA Rennes.

The goal of the project DECERT is to design and implement new efficient cooperating decision procedures (in particular for fragments of arithmetics), to standardize output interfaces based on certificates proof objects and to integrate SMT provers with skeptical proof assistants and larger verification contexts such as the Rodin tool for B and the Frama-C/Jessie tool chain for verifying C programs.

The partners are: CEA List, LORIA/INRIA Nancy - Grand Est, IRISA/INRIA Rennes - Bretagne Atlantique, INRIA Sophia Antipolis - Méditerranée, Systerel

8.3. European initiatives

8.3.1. European COST action FVOOS

Participants: Claude Marché [contact], Romain Bardou, François Bobot, Asma Tafat.

FVOOS (Formal Verification of Object-Oriented Programs, http://www.cost-ic0701.org/) is a COST (European Cooperation in the field of Scientific and Technical Research, http://www.cost.esf.org/) action. It started in 2008 and will last until April 2011.

It involves 40 academic groups among 18 countries in Belgium, Denmark, Estonia, France, Germany, Ireland, Israel, Italy, The Netherlands, New Zealand, Norway, Poland, Portugal, Romania, Spain, Sweden, Switzerland and United Kingdom.

The aim of this action is to develop verification technology with the reach and power to assure dependability of object-oriented programs on industrial scale.

8.4. Exterior research visitors

8.4.1. Visits

- M. Pouzet visited TheMathworks (Natick, USA) in July 2010 and gave a talk on VeLus, a formally certified Lustre compiler.

8.4.2. Invitations

- Thierry Coquand (University of Gothenburg, Sweden) visited our team in January as part of the Digiteo invitation program. He worked with C. Paulin and other researchers from Typical and MSR-INRIA research center on the use of the *Coq* proof assistant for the development of formal mathematics.
- Simão Melo de Sousa (Universidade da Beira Interior, Portugal) visited ProVal from September to November 2010. He worked with J.-C. Filliâtre on the deductive verification of ARM7 assembly programs, with application to the WCET problem.

9. Dissemination

9.1. Interaction with the scientific community

9.1.1. Prices and distinctions

Since 2007, Marc Pouzet is a junior member of the IUF (*"Institut Universitaire de France"*), that distinguishes each year a few French university professors for the high quality of their research activities.

9.1.2. Collective responsibilities within INRIA

- S. Boldo, elected representative of the researchers at the "comité de centre" of the INRIA Saclay - Île-de-France (2008–2010).
- S. Boldo, member of the CLHS, *comité local hygiène et sécurité* and member of the CLFP, *comité local de formation permanente*.
- S. Boldo, member of hiring committee of a communication engineer (IR-COM1) for the INRIA.
- C. Paulin, *déléguée scientifique* of the centre INRIA Saclay - Île-de-France and a member of the national evaluation board of INRIA until November.
- C. Paulin, member of hiring committee of DR2 national positions and CR2 positions at INRIA Nancy-Lorraine.

9.1.3. Collective responsibilities outside INRIA

- C. Paulin, director of the Graduate school in Computer Science at University Paris Sud http://dep-info.u-psud.fr/ed/.
- G. Melquiond, elected officer of the IEEE-1788 standardization committee on interval arithmetic since 2008.
- C. Marché, French National Coordinator for the COST action "Formal Verification of Object-Oriented Programs" (2008-2011).
- C. Marché (since April 2007) and C. Paulin (since Sep. 2010) , members of the program committee of Digiteo Labs, the world-class research park in *Île-de-France* region dedicated to information and communication science and technology, http://www.digiteo.fr/, since April 2007.
- C. Marché, member of the selection committee of the "DIM Logiciels et Systèmes Complexes", providing grants to research projects, funded by *Île-de-France* regional council and Digiteo cluster, http://www.dimlsc.fr/.
- E. Contejean and C. Marché, nominated members of the *"conseil du laboratoire"* of LRI since April 2010.
- C. Marché, hiring committee of one assistant professor position at IUT Orsay.
- C. Marché, M. Pouzet and X. Urbain: hiring committee for one assistant professor position at ENSIIE, in Evry (spring 2010).
- M. Pouzet, hiring committee for a full professor position at Ecole Supérieure d'Informatique et Applications de Lorraine (ESIAL) in Nancy (spring 2010).
- G. Melquiond, C. Paulin, members of the *"commission consultative de spécialistes de l'université"*, Section 27, University Paris-Sud 11 since April 2010.
- C. Paulin, hiring committees of two assistant professor positions: University Paris-Sud (chaire INRIA), and University Paris 7.
- C. Paulin, representative of Univ. Paris-Sud 11 for the education part of the EIT KIC ICT Labs.

- X. Urbain, hiring committee for two assistant professor positions at UPB/ENSEIRB-MATMECA, Bordeaux (Spring 2010).
- J.-C. Filliâtre is *correcteur au concours d'entrée à l'École Polytechnique* (computer science examiner for the entrance exam at École Polytechnique) since 2008.
- S. Conchon was *correcteur au concours d'entrée á l'École Polytechnique* (computer science examiner for the entrance exam at École Polytechnique) in 2010.
- É. Contejean is a member of the *"jury de l'agrégation externe de mathématiques"* as an expert in computer science since 2007.
- X. Urbain is an elected member of the board (*"conseil d'administration"*) of École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise (ENSIIE).

9.1.4. Event organization

- C. Marché co-organized (with B. Beckert, Karlsruhe Institute of Technology, Germany) the International Conference on Formal Verification of Object-Oriented Software (Paris, France, June 28-30, 2010). http://foveoos2010.cost-ic0701.org/ [37].
- J.-C. Filliâtre co-organized (with Cormac Flanagan, University of California, Santa Cruz, CA, USA) the PLPV'10 workshop (January 2010, Madrid, Spain). http://slang.soe.ucsc.edu/plpv10/.

9.1.5. Editorial boards

- S. Boldo is member of the editorial committee of the popular science web site)i(: http://interstices.info/.
- C. Paulin edited a special issue of Science of Computer Programming devoted to selected papers of the conference MPC'08 .
- Marc Pouzet is associate editor of the EURASIP Journal on Embedded systems (http://www.hindawi.com/journals/es/. He is "directeur de collection" for Hermes publisher.

9.1.6. Program committees

- C. Marché, program co-chair of the International Conference on Formal Verification of Object-Oriented Software (FoVeOOS 2010).
- S. Conchon, co-chair of the JFLA 2010 and program chair of JFLA 2011.
- S. Boldo is a member of the program committee of JFLA 2011.
- É. Contejean is a member of the program committee of PEPM 2011.
- C. Paulin is a member of the program committees of the 10th International Conference on Mathematics of Program Construction (MPC 2010), the second conference on Interactive Theorem Proving (ITP 2011), and the Fifth ACM SIGPLAN Workshop on Programming Languages meets Program Verification (PLPV 2011), affiliated to POPL.
- M. Pouzet is a member of the program committee of the following conferences in 2010: 31st IEEE Real-Time Systems Symposium conference (RTSS); Formal Methods in Computer Aided Design (FMCAD); Design, Automation & Test in Europe (DATE); Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL); Conference on Real-Time and Network Systems (RTNS) and the workshop of Design Correct Circuits (DCC), affiliated to ETAPS.
- J.-C. Filliâtre was a member of the program committees of AFM 2010, IWS 2010, Inforum 2010 and PLPV 2010.

9.1.7. Invited Presentations

- S. Boldo, invited speaker at the Third International Workshop on Numerical Software Verification (NVS-3) on July 15th 2010: *Formal verification of numerical programs: from C annotated programs to Coq proofs*.

9.1.8. Theses defended

- X. Urbain, Habilitation thesis, November 29th, 2010 [13].
- F. Plateau, PhD thesis, January 6th, 2010 [12]. She has been hired in the new company Prove & Run headed by D. Bolignano.
- J. Kanig, PhD thesis, November 26th, 2010. He was supervised by Jean-Christophe Filliâtre and Christine Paulin. In 2011, he will be software engineer at the AdaCore Company, at the European Office in Paris. (http://www.adacore.com).

9.1.9. Participation to thesis juries

- C. Marché: reviewer, PhD jury of Arthur Charguéraud (University Paris 7, December 16th, 2010).
- C. Marché: member of Habilitation jury of Xavier Urbain (University Paris 11, November 29th, 2010).
- C. Paulin: reviewer of the following PhD thesis: Benoït Robillard (CNAM, Nov 30th) and Santiago Zanella (Mines-Paristech, Dec 9th); member of Cédric Saule PhD thesis jury (Univ. Paris 11, Dec 17th, 2010).
- C. Paulin: member of Habilitation jury of Iordanis Kerenidis (University Paris 11, Dec 3th, 2010).
- M. Pouzet: reviewer of the following PhD thesis: Tayeb Bouhadiba (Université de Grenoble; dir: Florence Maraninchi; September 16th, 2010); Daniel Reynaud (INPL, Nancy; dir: Jean-Yves Marion; October 15th, 2010); Antony Coadou (Université de Nice; dir: Robert de Simone; December 3th, 2010).
- M. Pouzet: president of the PhD jury of Tayeb Bouhadiba and of the Habilitation jury of David Delahaye (December 9th, 2010).
- J.-C. Filliâtre: member of the PhD jury of Ioana Paşca (Université de Nice, November 23rd, 2010).

9.2. Teaching

9.2.1. Supervision of PhDs

- S. Boldo, C. Marché: Ph.D. thesis of T. Nguyen, started in February 2009, part of the Hisseo project (static analysis of the assembly code).
- S. Conchon, É. Contejean: Ph.D. thesis of Stéphane Lescuyer, defence scheduled January 2011 (complete certification of an automated theorem prover dedicated to program verification).
- S. Conchon, É. Contejean: Ph.D thesis of Mohamed Iguernelala, started September 2009 (forward and backward strategies in SMT solvers).
- J.-C. Filliâtre: Ph.D. thesis of Johannes Kanig, defended on November 26th, 2010
- J.-C. Filliâtre: Ph.D. thesis of François Bobot (started September 2008).
- C. Marché: PhD thesis of Romain Bardou since Sep. 07 (modular reasoning on pointer programs)
- C. Marché: PhD thesis of Asma Tafat since Sep. 09 (dynamic invariants)
- C. Marché, jointly with Benjamin Monate (CEA): Paolo Herms, since Oct. 08 (certification of Frama-C/Jessie/Why tool-chain).
- C. Paulin: Wendi Urribarrí (towards certified libraries) since Nov. 2006

- M. Pouzet: PhD thesis of Léonard Gérard since Sept 2008 (a language for n-synchronous systems) and Cédric Auger, started in Sept 2008 (certified compilation of Lustre).
- M. Pouzet was supervising the PhD of Florence Plateau (the theory of n-synchronous systems) which was defended January, 6th 2010.
- X. Urbain is (co-)supervising the PhD theses of A. Compaore (with P. Le Gall, on rewriting techniques for (space and time) simulation of biological processes), and of Z. Bouzid (with S. Tixeuil and M. Gradinariu Potop-Butucaru, on models and algorithms for emerging systems).

9.2.2. Supervision of Post-docs and internships

- C. Marché supervises the post-doc intern of K. Krishnamani since Sep 09 (predicate abstraction techniques for critical C programs).
- C. Paulin supervises together with P. Courtieu (CNAM) the post-doc intern of D. Baelde on certification of security of watermarking algorithms.
- J.-C. Filliâtre and A. Paskevich supervised the internship of S. Cruanes on integration of TPTP provers into the Why platform.
- M. Pouzet supervised the master internship of C. Pasteur, on the optimisation of the handling of arrays in the Scade compiler.

9.2.3. Graduate courses

- Master Parisien de Recherche en Informatique (MPRI) http://mpri.master.univ-paris7.fr/

  C. Paulin is responsible for the course “Proof assistants”. In 2010, she lectured (6h) in this course.

  In 2010, G. Melquiond lectured (9h) in the course on “Proof assistants”.

  In 2010-2011, X. Urbain lectured (12,5h) in the course on “Automated Deduction”.

  In 2010-2011, É. Contejean lectured on advanced rewriting (12h) in the course on “Automated Deduction”.

9.2.4. Other Courses

- L. Mandel, C. Paulin and M. Pouzet: complete professor duty (192h per year) at University Paris-Sud 11 and ENS.
- S. Conchon is teaching as part of his duty (96h per year) at University Paris-Sud 11.
- A. Paskevich: young associate professor (150h during academic year 2010/2011) at IUT d’Orsay, University Paris-Sud 11.
- In fall 2010, J.-C. Filliâtre is lecturing (24h) at École Normale Supérieure on programming languages and compilers. In 2009–2010, J.-C. Filliâtre is teaching at École Polytechnique (70h per year).
- C. Auger, R. Bardou, F. Bobot and L. Gérard: “moniteur” position (64h per year) at University Paris-Sud 11.
- A. Tafat, M. Iguernelala: “moniteur” position (64h per year) at I.U.T Orsay.
- T. Nguyen: “moniteur” position (64h per year) at IFIPS (since September 2009)

9.3. Industrial Dissemination

- The tools Frama-C, Why and Alt-Ergo were presented at the Imatch day on 23 november 2010, on the themes of security and proof of programs (C. Marché, S. Conchon, C. Paulin) http://www.inria.fr/centres-de-recherche-inria/saclay-ile-de-france/agenda/imatch-securite-preuve-de-programmes

- Airbus France expressed in 2009 the wish to integrate our tool Alt-Ergo in its process of certification of the critical softwares in their next generation planes. We thus started the procedure of *qualifying* Alt-Ergo in the sense of the DO-178B norm, which fixes the contraints on software development to achieve certification of an avionics software. This is done as part of the *ADT Alt-Ergo*.
- Journees INRIA-Industrie in Toulouse: Sylvain Conchon presented a demo of the Alt-Ergo theorem prover.
- G. Melquiond participates in the meetings of the IEEE-1788 standardization committee on interval arithmetic. The "Technology Development" INRIA department is funding his travel expenses till late 2011.
- The Frama-C environment has a growing industrial impact, being currently under experimental use and evaluation by U3CAT industrial partners but also other european industrial users, e.g. Fraunhofer FIRST institute (http://www.first.fraunhofer.de/) in Berlin, Germany; Bosch company in Germany; Thalès group, France (within european project ITEA2 SPICES http://www.spices-itea.org).

9.4. Popularization

Since April 2008, S. Boldo is member of the editorial committee of the popular science web site)i(: http://interstices.info/.

Since July 2009, S. Boldo is elected member of the board of the Animath association that promotes mathematics among young people.

S. Boldo was invited to talk at Intertice on May 10th 2010, a meeting for teachers about teaching and TICE. She talked about the seminar "Formation Informatique et Objets Numériques", where she was speaker, which was organized in 2009 in order to prepare a computer science option in the secondary schools of the academy of Versailles.

S. Boldo gave a talk for mathematic secondary school teachers at the IUFM on June 3rd 2010.

S. Boldo wrote an article for the popular science web site)i(about the fact that it is always the computer's fault: http://interstices.info/idee-recue-informatique-18 [41].

S. Boldo was invited to a participate to the web-TV of the Cité des Sciences et de l'Industrie for the show *Qui veut gagner des neurones?* about computer science: http://www.universcience.tv/media/1340/l-informatique.html [42].

R. Bardou, F. Bobot, M. Iguernelala: *"Salon de la Culture et des Jeux Mathématiques"*, Paris, May 27-30. Animation of a workshop introducing the bases of programming, illustrated by the programmation of a robot. Jointly with Y. Régis-Gianas, pi.r2 team, INRIA Paris-Rocquencourt.

F. Bobot, M. Iguernelala, J. Kanig, A. Tafat: *"Fête de la science"*, Gif-sur-Yvette, October 22-24. Introduction to computer programming based on an improved presentation of the robot animation above.

10. Bibliography

Major publications by the team in recent years

[1] S. BOLDO. *Floats & Ropes: a case study for formal numerical program verification*, in "36th International Colloquium on Automata, Languages and Programming", Rhodos, Greece, Lecture Notes in Computer Science - ARCoSS, Springer, July 2009, vol. 5556, p. 91–102.

[2] S. BOLDO, J.-C. FILLIÂTRE. *Formal Verification of Floating-Point Programs*, in "18th IEEE International Symposium on Computer Arithmetic", Montpellier, France, June 2007, p. 187-194, http://www.lri.fr/~filliatr/ftp/publis/caduceus-floats.pdf.

[3] J.-L. COLAÇO, B. PAGANO, M. POUZET. *A Conservative Extension of Synchronous Data-flow with State Machines*, in "ACM International Conference on Embedded Software (EMSOFT'05)", Jersey city, New Jersey, USA, September 2005.

[4] É. CONTEJEAN, P. COURTIEU, J. FOREST, O. PONS, X. URBAIN. *Certification of automated termination proofs*, in "6th International Symposium on Frontiers of Combining Systems (FroCos 07)", Liverpool,UK, B. KONEV, F. WOLTER (editors), Lecture Notes in Artificial Intelligence, Springer, September 2007, vol. 4720, p. 148–162.

[5] É. CONTEJEAN, C. MARCHÉ, A. P. TOMÁS, X. URBAIN. *Mechanically proving termination using polynomial interpretations*, in "Journal of Automated Reasoning", 2005, vol. 34, n° 4, p. 325–363, http://dx.doi.org/10.1007/s10817-005-9022-x.

[6] J.-C. FILLIÂTRE. *Verification of Non-Functional Programs using Interpretations in Type Theory*, in "Journal of Functional Programming", July 2003, vol. 13, n° 4, p. 709–745, http://www.lri.fr/~filliatr/ftp/publis/jphd.pdf.

[7] J.-C. FILLIÂTRE, C. MARCHÉ. *Multi-Prover Verification of C Programs*, in "6th International Conference on Formal Engineering Methods", Seattle, WA, USA, J. DAVIES, W. SCHULTE, M. BARNETT (editors), Lecture Notes in Computer Science, Springer, November 2004, vol. 3308, p. 15–29, http://www.lri.fr/~filliatr/ftp/publis/caduceus.ps.gz.

[8] T. HUBERT, C. MARCHÉ. *A case study of C source code verification: the Schorr-Waite algorithm*, in "3rd IEEE International Conference on Software Engineering and Formal Methods (SEFM'05)", Koblenz, Germany, B. K. AICHERNIG, B. BECKERT (editors), IEEE Comp. Soc. Press, September 2005, http://www.lri.fr/~marche/hubert05sefm.ps.

[9] S. KRSTIĆ, S. CONCHON. *Canonization for disjoint unions of theories*, in "Information and Computation", May 2005, vol. 199, n° 1-2, p. 87–106.

[10] C. MARCHÉ, C. PAULIN-MOHRING, X. URBAIN. *The* KRAKATOA *Tool for Certification of* JAVA/JAVACARD *Programs annotated in* JML, in "Journal of Logic and Algebraic Programming", 2004, vol. 58, n° 1–2, p. 89–106, http://krakatoa.lri.fr.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] J. KANIG. *Spécification et preuve de programmes d'ordre supérieur*, Université Paris-Sud, 2010.

[12] F. PLATEAU. *Modèle n-synchrone pour la programmation de réseaux de Kahn à mémoire bornée*, Université Paris-Sud, 2010.

[13] X. URBAIN. *Preuve automatique : techniques, outils et certification*, Université Paris-Sud 11, November 2010, Thèse d'habilitation.

Articles in International Peer-Reviewed Journal

[14] S. BOLDO, J.-M. MULLER. *Exact and Approximated error of the FMA*, in "IEEE Transactions on Computers", 2010, http://hal.inria.fr/inria-00429617/en/.

[15] M. DAUMAS, G. MELQUIOND. *Certification of bounds on expressions involving rounded operators*, in "Transactions on Mathematical Software", 2010, vol. 37, n^{o} 1, http://hal.archives-ouvertes.fr/inria-00534350/fr/.

[16] Y. MOY, C. MARCHÉ. *Modular Inference of Subprogram Contracts for Safety Checking*, in "Journal of Symbolic Computation", 2010, vol. 45, p. 1184-1211, http://hal.inria.fr/inria-00534331/en/.

[17] F. DE DINECHIN, C. LAUTER, G. MELQUIOND. *Certifying the floating-point implementation of an elementary function using Gappa*, in "IEEE Transactions on Computers", 2010, p. 1–14, http://hal.inria.fr/inria-00533968/en/.

Invited Conferences

[18] S. BOLDO. *Formal verification of numerical programs: from C annotated programs to Coq proofs*, in "Proceedings of the Third International Workshop on Numerical Software Verification", Edinburgh, Scotland, July 2010, http://hal.inria.fr/inria-00534400/en/.

International Peer-Reviewed Conference/Proceedings

[19] A. AYAD, C. MARCHÉ. *Multi-Prover Verification of Floating-Point Programs*, in "Fifth International Joint Conference on Automated Reasoning", Edinburgh, Scotland, J. GIESL, R. HÄHNLE (editors), Lecture Notes in Artificial Intelligence, Springer, July 2010, http://www.lri.fr/~marche/ayad10ijcar.pdf.

[20] M. BARBOSA, J.-C. FILLIÂTRE, J. S. PINTO, B. VIEIRA. *A Deductive Verification Platform for Cryptographic Software*, in "4th International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2010)", Pisa, Italy, September 2010.

[21] A. BENVENISTE, B. CAILLAUD, M. POUZET. *The Fundamentals of Hybrid Systems Modelers*, in "49th IEEE International Conference on Decision and Control (CDC)", Atlanta, Georgia, USA, December 15-17 2010.

[22] S. BOLDO, F. CLÉMENT, J.-C. FILLIÂTRE, M. MAYERO, G. MELQUIOND, P. WEIS. *Formal Proof of a Wave Equation Resolution Scheme: the Method Error*, in "Proceedings of the first Interactive Theorem Proving Conference", Edinburgh, Scotland, M. KAUFMANN, L. C. PAULSON (editors), LNCS, Springer, July 2010, vol. 6172, p. 147–162, http://hal.inria.fr/inria-00450789/en/.

[23] S. BOLDO, T. M. T. NGUYEN. *Hardware-independent proofs of numerical programs*, in "Proceedings of the Second NASA Formal Methods Symposium", Washington D.C., USA, C. MUÑOZ (editor), NASA Conference Publication, April 2010, p. 14–23, http://hal.inria.fr/inria-00534410/en/.

[24] A. CIMATTI, A. FRANZEN, A. GRIGGIO, K. KALYANASUNDARAM, M. ROVERI. *Tighter Integration of BDDs and SMT for Predicate Abstraction*, in "Design, Automation & Test in Europe", Dresden. Germany, IEEE, March 2010.

[25] É. CONTEJEAN, P. COURTIEU, J. FOREST, A. PASKEVICH, O. PONS, X. URBAIN. *A3PAT, an Approach for Certified Automated Termination Proofs*, in "Partial Evaluation and Program Manipulation", Madrid, Spain, ACM Press, january 2010.

[26] L. MANDEL, F. PLATEAU, M. POUZET. *Clock Typing of n-Synchronous Programs*, in "Designing Correct Circuits (DCC 2010)", Paphos, Cyprus, March 2010.

[27] L. MANDEL, F. PLATEAU, M. POUZET. *Lucy-n: a n-Synchronous Extension of Lustre*, in "Tenth International Conference on Mathematics of Program Construction (MPC 2010)", Québec, Canada, June 2010, http://www.lri.fr/~mandel/papiers/MandelPlateauPouzet-MPC-10.pdf.

[28] A. TAFAT, S. BOULMÉ, C. MARCHÉ. *A Refinement Methodology for Object-Oriented Programs*, in "Formal Verification of Object-Oriented Software, Papers Presented at the International Conference", Paris, France, B. BECKERT, C. MARCHÉ (editors), Karlsruhe Reports in Informatics, June 2010, p. 143–159.

[29] E. TUSHKANOVA, A. GIORGETTI, C. MARCHÉ, O. KOUCHNARENKO. *Specifying Generic Java Programs: two case studies*, in "Tenth Workshop on Language Descriptions, Tools and Applications", C. BRABRAND, P.-E. MOREAU (editors), ACM Press, 2010, http://hal.inria.fr/inria-00525784/en/.

National Peer-Reviewed Conference/Proceedings

[30] R. BARDOU, C. MARCHÉ. *Perle de preuve: les tableaux creux*, in "Vingt-deuxièmes Journées Francophones des Langages Applicatifs", La Bresse, France, INRIA, January 2011.

[31] S. CONCHON, J.-C. FILLIÂTRE, F. LE FESSANT, J. ROBERT, G. VON TOKARSKI. *Observation temps-réel de programmes Caml*, in "Vingt-et-unièmes Journées Francophones des Langages Applicatifs", Vieux-Port La Ciotat, France, INRIA, January 2010, http://www.lri.fr/~conchon/publis/ocamlviz-jfla2010.pdf.

[32] J.-C. FILLIÂTRE, K. KALYANASUNDARAM. *Une bibliothèque de calcul distribué pour Objective Caml*, in "Vingt-deuxièmes Journées Francophones des Langages Applicatifs", La Bresse, France, INRIA, January 2011.

[33] L. MANDEL. *Cours de ReactiveML*, in "Vingt-et-unièmes Journées Francophones des Langages Applicatifs", Vieux-Port La Ciotat, France, INRIA, January 2010, http://www.lri.fr/~mandel/papiers/Mandel-JFLA-2010.pdf.

[34] L. MANDEL, F. PLATEAU, M. POUZET. *Lucy-n : une extension n-synchrone de Lustre*, in "Vingt-et-unièmes Journées Francophones des Langages Applicatifs", Vieux-Port La Ciotat, France, INRIA, January 2010, http://www.lri.fr/~mandel/papiers/MandelPlateau-JFLA-2010.pdf.

Workshops without Proceedings

[35] P. HERMS. *Certification of a chain for deductive program verification*, in "2nd Coq Workshop, satellite of ITP'10", Y. BERTOT (editor), 2010.

Scientific Books (or Scientific Book chapters)

[36] J.-M. MULLER, N. BRISEBARRE, F. DE DINECHIN, C.-P. JEANNEROD, V. LEFÈVRE, G. MELQUIOND, N. REVOL, D. STEHLÉ, S. TORRES. *Handbook of Floating-Point Arithmetic*, Birkhäuser, 2010.

Books or Proceedings Editing

[37] B. BECKERT, C. MARCHÉ (editors). *Formal Verification of Object-Oriented Software, Papers Presented at the International Conference*, Karlsruhe Reports in Informatics, June 2010.

Research Reports

[38] R. BARDOU, C. MARCHÉ. *Regions and Permissions for Verifying Data Invariants*, INRIA, 2010, n° RR-7412, http://hal.inria.fr/inria-00525384/en/.

[39] A. PASKEVICH. *Algebraic types and pattern matching in the logical language of the Why verification platform (version 2)*, INRIA, 2010, n° RR-7128, http://hal.inria.fr/inria-00439232/en/.

[40] A. TAFAT, S. BOULMÉ, C. MARCHÉ. *A Refinement Approach for Correct-by-Construction Object-Oriented Programs*, INRIA, 2010, n° RR-7310, http://hal.inria.fr/inria-00491835/en/.

Scientific Popularization

[41] S. BOLDO. *C'est la faute à l'ordinateur!*, February 2010, Interstices – Idée reçue, http://hal.inria.fr/inria-00534848/fr/.

[42] S. BOLDO. *L'informatique*, April 2010, Universcience web television, http://hal.inria.fr/inria-00534852/fr/.

[43] T. M. T. NGUYEN, S. BOLDO, C. MARCHÉ. *Formal proofs of numerical programs*, October 2010, Poster at the Digiteo Forum, Palaiseau, France, http://hal.inria.fr/inria-00536135/en/.

References in notes

[44] *The MAUDE System*.

[45] J. ANDRONICK. *Modélisation et vérification formelles de systèmes embarqués dans les cartes à microprocessur. Plateforme Java Card et Système d'exploitation*, Université Paris-Sud, March 2006, http://jandronick.free.fr/publi/these_june_andronick.pdf.

[46] J. ANDRONICK, B. CHETALI, C. PAULIN-MOHRING. *Formal Verification of Security Properties of Smart Card Embedded Source Code*, in "International Symposium of Formal Methods Europe (FM'05)", Newcastle,UK, J. FITZGERALD, I. J. HAYES, A. TARLECKI (editors), Lecture Notes in Computer Science, Springer, July 2005, vol. 3582, http://jandronick.free.fr/publi/FM2005.pdf.

[47] T. ARTS, J. GIESL. *Termination of term rewriting using dependency pairs*, in "Theoretical Computer Science", 2000, vol. 236, p. 133–178.

[48] P. AUDEBAUD, C. PAULIN-MOHRING. *Proofs of Randomized Algorithms in Coq*, in "Science of Computer Programming", 2009, vol. 74, n° 8, p. 568–589, http://hal.inria.fr/inria-00431771/en/.

[49] R. BARDOU, J.-C. FILLIÂTRE, J. KANIG, S. LESCUYER. *Faire bonne figure avec Mlpost*, in "Vingtièmes Journées Francophones des Langages Applicatifs", Saint-Quentin sur Isère, INRIA, January 2009, http://www.lri.fr/~filliatr/ftp/publis/mlpost-fra.pdf.

[50] B. BARRAS. *Verification of the Interface of a Small Proof System in Coq*, in "Types for Proofs and Programs, International Workshop TYPES'96, Aussois, France, December 15-19, 1996, Selected Papers", E. GIMÉNEZ, C. PAULIN-MOHRING (editors), Lecture Notes in Computer Science, Springer, 1998, vol. 1512, p. 28-45.

[51] G. BARTHE, B. GRÉGOIRE, S. Z. BÉGUELIN. *Formal certification of code-based cryptographic proofs*, in "POPL", Savannah, GA, USA, Z. SHAO, B. C. PIERCE (editors), ACM Press, January 2009, p. 90-101.

[52] P. BAUDIN, J.-C. FILLIÂTRE, C. MARCHÉ, B. MONATE, Y. MOY, V. PREVOSTO. *ACSL: ANSI/ISO C Specification Language*, 2008, http://frama-c.cea.fr/acsl.html.

[53] A. BENVENISTE, P. CASPI, S. A. EDWARDS, N. HALBWACHS, P. LE GUERNIC, R. DE SIMONE. *The synchronous languages 12 years later*, in "Proceedings of the IEEE", January 2003, vol. 91, n^{o} 1.

[54] G. BERRY, G. GONTHIER. *The Esterel synchronous programming language, design, semantics, implementation*, in "Science of Computer Programming", 1992, vol. 19, n^{o} 2, p. 87-152.

[55] Y. BERTOT, N. MAGAUD, P. ZIMMERMANN. *A Proof of GMP Square Root*, in "Journal of Automated Reasoning", 2002, vol. 29, n^{o} 3-4, p. 225–252.

[56] D. BIERNACKI, J.-L. COLAÇO, G. HAMON, M. POUZET. *Clock-directed Modular Code Generation of Synchronous Data-flow Languages*, in "ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)", Tucson, Arizona, June 2008.

[57] D. BIERNACKI, J.-L. COLAÇO, M. POUZET. *Clock-directed Modular Code Generation from Synchronous Block Diagrams*, in "Workshop on Automatic Program Generation for Embedded Systems (APGES 2007)", Salzburg, Austria, October 2007, http://www-fp.dcs.st-and.ac.uk/APGES/OnlineProceedings/11-Pouzet.pdf.

[58] S. BOLDO. *Pitfalls of a full floating-point proof: example on the formal proof of the Veltkamp/Dekker algorithms*, in "Third International Joint Conference on Automated Reasoning", Seattle, USA, U. FURBACH, N. SHANKAR (editors), Lecture Notes in Computer Science, Springer, August 2006, vol. 4130, p. 52-66, http://www.springerlink.com/content/524v5246177t0877/.

[59] R. BORNAT. *Proving Pointer Programs in Hoare Logic*, in "Mathematics of Program Construction", 2000, p. 102–126.

[60] P. CASPI, M. POUZET. *Synchronous Kahn Networks*, in "ACM SIGPLAN International Conference on Functional Programming", Philadelphia, Pensylvania, May 1996, http://portal.acm.org/citation.cfm?id=232651.

[61] P. CASPI, M. POUZET. *A Co-iterative Characterization of Synchronous Stream Functions*, in "Coalgebraic Methods in Computer Science (CMCS'98)", Electronic Notes in Theoretical Computer Science, March 1998.

[62] V. CHAUDHARY. *The Krakatoa tool for certification of Java/JavaCard programs annotated in JML : A Case Study*, IIT internship report, July 2004.

[63] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *Synchronizing Periodic Clocks*, in "ACM International Conference on Embedded Software (EMSOFT'05)", Jersey city, New Jersey, USA, September 2005.

[64] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *N-Synchronous Kahn Networks: a Relaxed Model of Synchrony for Real-Time Systems*, in "ACM International Conference on Principles of Programming Languages (POPL'06)", Charleston, South Carolina, USA, January 2006.

[65] A. COHEN, L. MANDEL, F. PLATEAU, M. POUZET. *Abstraction of Clocks in Synchronous Data-flow Systems*, in "The Sixth ASIAN Symposium on Programming Languages and Systems (APLAS)", Bangalore, India, December 2008, http://www.lri.fr/~plateau/papers/aplas08.pdf.

[66] J.-L. COLAÇO, G. HAMON, M. POUZET. *Mixing Signals and Modes in Synchronous Data-flow Systems*, in "ACM International Conference on Embedded Software (EMSOFT'06)", Seoul, South Korea, October 2006.

[67] J.-L. COLAÇO, M. POUZET. *Clocks as First Class Abstract Types*, in "Third International Conference on Embedded Software (EMSOFT'03)", Philadelphia, Pennsylvania, USA, October 2003.

[68] J.-L. COLAÇO, M. POUZET. *Type-based Initialization Analysis of a Synchronous Data-flow Language*, in "International Journal on Software Tools for Technology Transfer (STTT)", August 2004, vol. 6, n^{o} 3, p. 245–255.

[69] S. CONCHON, S. KRSTIĆ. *Strategies for Combining Decision Procedures*, in "Theoretical Computer Science", 2006, vol. 354, n^{o} 2, p. 187–210.

[70] É. CONTEJEAN, P. CORBINEAU. *Reflecting Proofs in First-Order Logic with Equality*, in "20th International Conference on Automated Deduction (CADE-20)", Tallinn, Estonia, R. NIEUWENHUIS (editor), Lecture Notes in Artificial Intelligence, Springer, July 2005, vol. 3632, p. 7–22.

[71] É. CONTEJEAN, P. COURTIEU, J. FOREST, O. PONS, X. URBAIN. *Certification of automated termination proofs*, CEDRIC, May 2007, n^{o} 1185.

[72] J.-F. COUCHOT, S. LESCUYER. *Handling Polymorphism in Automated Deduction*, in "21th International Conference on Automated Deduction (CADE-21)", Bremen, Germany, LNCS (LNAI), July 2007, vol. 4603, p. 263–278, http://www.lri.fr/~lescuyer/pdf/talkCADE.pdf.

[73] F. DURÁN, S. LUCAS, J. MESEGUER, C. MARCHÉ, X. URBAIN. *Proving Termination of Membership Equational Programs*, in "ACM SIGPLAN 2004 Symposium on Partial Evaluation and Program Manipulation", Verona, Italy, ACM Press, August 2004.

[74] J.-C. FILLIÂTRE. *Formal Proof of a Program: Find*, in "Science of Computer Programming", 2006, vol. 64, p. 332–240, http://www.lri.fr/~filliatr/ftp/publis/find.pdf.

[75] J.-C. FILLIÂTRE, S. OWRE, H. RUESS, N. SHANKAR. *ICS: Integrated Canonization and Solving (Tool presentation)*, in "Proceedings of CAV'2001", G. BERRY, H. COMON, A. FINKEL (editors), Lecture Notes in Computer Science, Springer, 2001, vol. 2102, p. 246–249.

[76] J. GERLACH, J. BURGHARDT. *An Experience Report on the Verification of Algorithms in the C++ Standard Library using Frama-C*, in "Formal Verification of Object-Oriented Software, Papers Presented at the International Conference", Paris, France, B. BECKERT, C. MARCHÉ (editors), Karlsruhe Reports in Informatics, June 2010, p. 191–204.

[77] B. GRAMLICH. *On Proving Termination by Innermost Termination*, in "7th International Conference on Rewriting Techniques and Applications", New Brunswick, NJ, USA, H. GANZINGER (editor), Lecture Notes in Computer Science, Springer, July 1996, vol. 1103, p. 93–107.

[78] N. HALBWACHS, P. CASPI, P. RAYMOND, D. PILAUD. *The Synchronous Dataflow Programming Language* LUSTRE, in "Proceedings of the IEEE", September 1991, vol. 79, n^{o} 9, p. 1305-1320.

[79] T. HUBERT. *Analyse Statique et preuve de Programmes Industriels Critiques*, Université Paris-Sud, June 2008, http://www.lri.fr/~marche/hubert08these.pdf.

[80] T. HUBERT, C. MARCHÉ. *Separation Analysis for Deductive Verification*, in "Heap Analysis and Verification (HAV'07)", Braga, Portugal, March 2007, p. 81–93, http://www.lri.fr/~marche/hubert07hav.pdf.

[81] B. JACOBS, C. MARCHÉ, N. RAUCH. *Formal Verification of a Commercial Smart Card Applet with Multiple Tools*, in "Algebraic Methodology and Software Technology", Stirling, UK, Lecture Notes in Computer Science, Springer, July 2004, vol. 3116.

[82] K. R. M. LEINO, M. MOSKAL. *VACID-0: Verification of Ample Correctness of Invariants of Data-structures, Edition 0*, in "Proceedings of Tools and Experiments Workshop at VSTTE", 2010.

[83] X. LEROY. *Formal certification of a compiler back-end, or: programming a compiler with a proof assistant*, in "Conference Record of the 33rd Symposium on Principles of Programming Languages", Charleston, South Carolina, ACM Press, January 2006.

[84] P. LETOUZEY. *A New Extraction for Coq*, in "TYPES 2002", H. GEUVERS, F. WIEDIJK (editors), Lecture Notes in Computer Science, Springer, 2003, vol. 2646, http://www.springerlink.com/content/3114f6rp11b5qg24/.

[85] P. LETOUZEY. *Programmation fonctionnelle certifiée: l'extraction de programmes dans l'assistant Coq*, Université Paris-Sud, July 2004, http://tel.archives-ouvertes.fr/tel-00150912/en/.

[86] L. MANDEL, M. POUZET. *ReactiveML, a Reactive Extension to ML*, in "ACM International Conference on Principles and Practice of Declarative Programming (PPDP)", Lisboa, July 2005, p. 82–93, http://www.lri.fr/~mandel/papers/MandelPouzet-PPDP-2005.pdf.

[87] C. MARCHÉ, C. PAULIN-MOHRING. *Reasoning about Java Programs with Aliasing and Frame Conditions*, in "18th International Conference on Theorem Proving in Higher Order Logics", J. HURD, T. MELHAM (editors), Lecture Notes in Computer Science, Springer, August 2005, vol. 3603, p. 179–194, http://www.lri.fr/~marche/marche05tphols.ps.

[88] C. MARCHÉ, N. ROUSSET. *Verification of Java Card Applets Behavior with respect to Transactions and Card Tears*, in "4th IEEE International Conference on Software Engineering and Formal Methods (SEFM'06)", Pune, India, D. V. HUNG, P. PANDYA (editors), IEEE Comp. Soc. Press, September 2006, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1698731.

[89] C. MARCHÉ, X. URBAIN. *Modular and Incremental Proofs of AC-Termination*, in "Journal of Symbolic Computation", 2004, vol. 38, p. 873–897, http://authors.elsevier.com/sd/article/S074771710400029X.

[90] C. MARCHÉ. *Preuves mécanisées de Propriétés de Programmes*, Université Paris 11, December 2005, Thèse d'habilitation, http://www.lri.fr/~marche/marche05hdr.pdf.

[91] Y. MOY. *Automatic Modular Static Safety Checking for C Programs*, Université Paris-Sud, January 2009, http://www.lri.fr/~marche/moy09phd.pdf.

[92] E. OHLEBUSCH, C. CLAVES, C. MARCHÉ. *TALP: A Tool for the Termination Analysis of Logic Programs*, in "11th International Conference on Rewriting Techniques and Applications", Norwich, UK, L. BACHMAIR (editor), Lecture Notes in Computer Science, Springer, July 2000, vol. 1833, p. 270–273, http://theorie.informatik.uni-ulm.de/Personen/eo/PAPERS/RTA00.ps.gz.

[93] S. RANISE, C. TINELLI. *The Satisfiability Modulo Theories Library (SMT-LIB)*, 2006, http://www.smtcomp.org.

[94] M. SOZEAU. *Program-ing Finger Trees in Coq*, in "12th ACM SIGPLAN International Conference on Functional Programming, ICFP 2007", Freiburg, Germany, R. HINZE, N. RAMSEY (editors), ACM Press, 2007, p. 13–24, http://mattam.org/research/publications/Program-ing_Finger_Trees_in_Coq-icfp07-011007.pdf.

[95] M. SOZEAU. *Un environnement pour la programmation avec types dépendants*, Université Paris-Sud, December 2008.

[96] D. STEVENSON. *A proposed standard for binary floating point arithmetic*, in "IEEE Computer", 1981, vol. 14, n^{o} 3, p. 51-62.

[97] L. THÉRY. *Proving Pearl: Knuth's algorithm for prime numbers*, in "Proceedings of the 16th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2003)", D. BASIN, B. WOLFF (editors), LNCS, Springer-Verlag, 2003, vol. 2758.

[98] X. URBAIN. *Approche incrémentale des preuves automatiques de terminaison*, Université Paris-Sud, Orsay, France, October 2001, http://www.lri.fr/~urbain/textes/these.ps.gz.

[99] J. VUILLEMIN. *On Circuits and Numbers*, Digital, Paris Research Laboratory, 1993.